611-TD-556-001

EOSDIS Core System Project

M&O Procedures: Section 15 - Quality Assurance

Interim Update

February 2000

Raytheon Systems Company
Upper Marlboro, Maryland

Preface

This document is an interim update to the Mission Operations Procedures Manual for the ECS Project, document number 611-CD-550-001. This document has not been submitted to NASA for approval, and should be considered unofficial.

This document contains updated procedures for the QA Metadata Update Tool that is currently being integrated into the ECS baseline as a DAAC Unique Extension per ESD 70. It is subject to change when that work is completed for the formal release. The current update incorporated the *Instructions and Guide on Using the Database QA Updater 1.3.* Any questions should be addressed to Dr. Richard Buss or Rodney Creecy.

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5301

This page intentionally left blank.

15. Quality Assurance

This section describes the tools available for science data Quality Assurance (QA) - the QA Monitor and the QA Metadata Update Tool (QAMUT) .

Operational Quality Assurance is performed by DAAC operations personnel authorized to modify the value of the Operational QA flag attribute value for a product generated at the DAAC. The operator has the capability to view the product through EOSView and retrieve production history files. Figure 15-1 provides an overview of the quality assurance process.

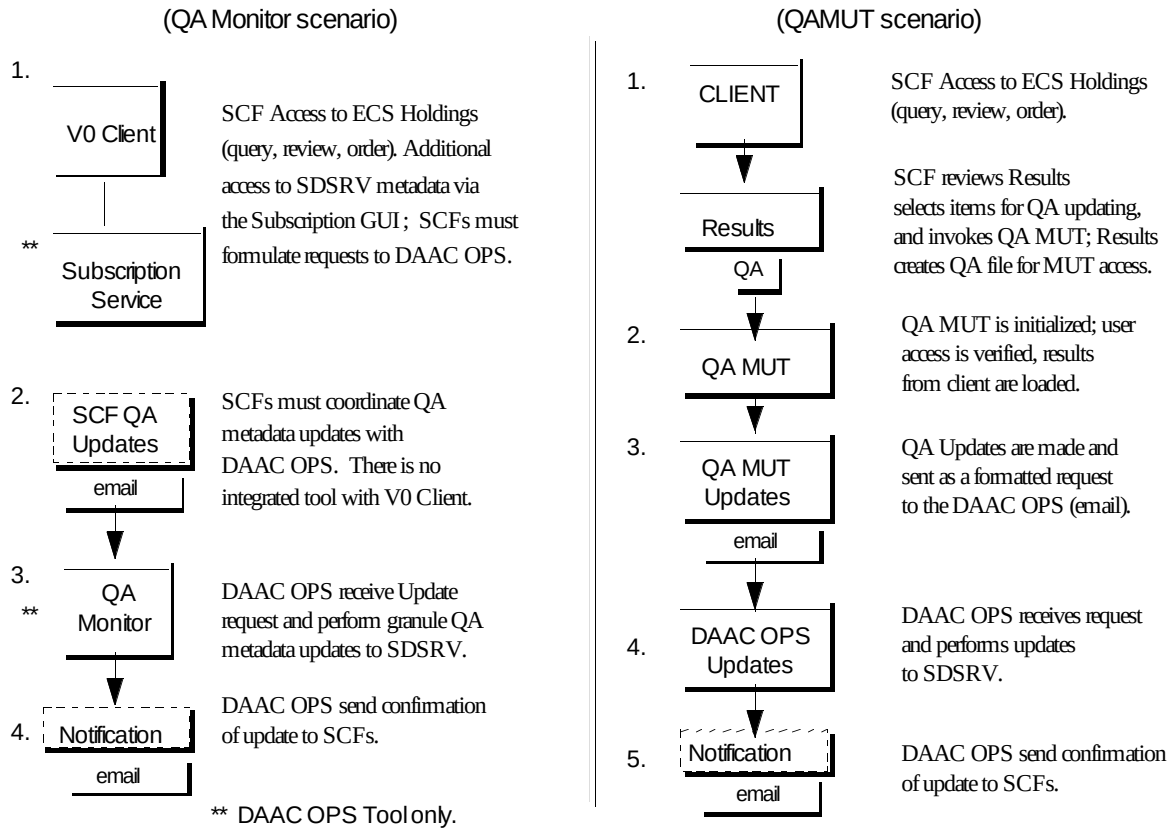

Figure 15-1. QA Metadata Update Process

15.1 QA Monitor

The purpose of the QA Monitor is to enable DAAC QA experts to modify ScienceQualityFlag and OperationalQualityFlag attributes of core metadata for a granule. SCFs send email requests

to the DAAC to update the ScienceQualityFlag attribute, until the QA Metadata Update Tool (Section 15.2) is available.

The QA Monitor can be used to request the Science Data Server to search for specific types of Data Granules; Query, Retrieve, and Update (QRU) Metadata; transfer Data Granules to the operator's computer; and transfer Production History to the operator's computer. It can also be used to update data granule metadata, view graphical images of data granules, and print/display lists of data granules and data types. Table 15.1-1 summarizes common operator functions performed with the QA Monitor. Table 15.1-2 describes the fields in the QA Monitor; Table 15.1-2 provides the usage for each of the pushbuttons.

The QA Monitor is launched by clicking on the Desktop Icon, or via Unix commands. The Unix commands are provided following Tables 15.1-1 through 15.1-3.

Table 15.1-1. Common Operator Functions Performed with the QA Monitor

Operator Function	Description	Purpose
Query Data Granules	Initiates a request to search the science archive for data granules	Find all archived data granules with the same data type which were inserted into the archive at a certain time (data interval)
Retrieve Data Granules	Initiates a request to get data granules from the science archive	Transfer data granule(s) from archive to local disk for visualization
Visualize Data (HDF files)	Display Visualize screen	View graphical image of data granules to assess quality
Update Metadata	Initiates a request to archive QA information about data granules	Update data granule QA information in the archive, based on DAAC QA activities encompassing use of the Visualize Data function.

Table 15.1-2. QA Monitor GUI Fields

Field Name	Data Type	Size	Entry	Description
Data Granule Insert Begin End	Date min max	 1/2/1901 6/1/2036	Initial default value - can be changed by user	Search criteria for granule metadata beginning and end date
Data Types list	single selection	N/A	User selects a data type from the list displayed at startup	The list of all available data types at a specific DAAC
Data Granules list	multiple selection	N/A	User clicks data granule row(s) then clicks retrieve pushbutton	The list of all data granules in the date interval above for a particular selected data type are available for retrieval.
Status	text	N/A	Displays status messages only	Displays status messages

Table 15.1-3. QA Monitor GUI Pushbuttons

Button	Description
Query	Populates list of data granules on the bottom half of the GUI for a particular selected data type within a data interval
Find	(below the data types and data granules list) - allow the operator to perform a keyword search for information stored in those 2 lists.
Retrieve	Allows the operator to retrieve data granule(s) or production history tar file(s) from the DAAC's data archive and place on the local disk.
Update	Pops up a Granule Parameters screen Update Metadata Dialog

Launching the QA Monitor Using UNIX Commands

1 Access the command shell.
The command shell prompt is displayed.

NOTE: Commands in Steps 2 through 14 are typed at a UNIX system prompt.

2 Type **xhost +** then press the **Return/Enter** key on the keyboard.

3 Open another UNIX window.

4 Start the log-in to the Data Processing Subsystem host by typing either **telnet *hostname*** (e.g., **g0sps06**), **rlogin *hostname***, or **rsh *hostname*** in the new window then press the **Return/Enter** key.

If you use the **telnet** command, a **Login:** prompt appears; continue with Step 5.
If you use either the **rlogin** or **rsh** command, the system uses the User ID currently in use; go to Step 6.

5 If a **Login:** prompt appears, log in as yourself by typing your ***UserID*** then pressing the **Return/Enter** key.

6 At the **Password:** prompt type your ***Password*** then press the **Return/Enter** key.

7 Start the log-in to DCE by typing **dce_login** then pressing the **Return/Enter** key.

8 At the **Enter Principal Name:** prompt type your ***DCE UserID*** then press the **Return/Enter** key.

9 At the **Enter Password:** prompt type your ***DCE Password*** then press the **Return/Enter** key.

10 Type **setenv DISPLAY *clientname*:0.0** then press the **Return/Enter** key.
Use either the terminal/workstation IP address or the machine-name for the ***clientname***.

11 Type **setenv MODE *mode*** then press the **Return/Enter** key.
The ***mode*** will most likely be one of the following operating modes:
OPS (for normal operation).
TS1 (for SSI&T).

TS2 (new version checkout).

Note that the separate subdirectories under /usr/ecs apply to (describe) different operating modes.

12 Type **source /usr/ecs/*mode*/CUSTOM/utilities/EcCoEnvCsh** then press **Return/Enter**.
The **source** command sets the environment variables identified in the specified file.

13 Type **cd /*path*** then press **Return/Enter**.
Change directory to the directory (e.g., /usr/ecs/*mode*/CUSTOM/utilities) containing the data processing start scripts (e.g., EcDpPrStartQaMonitorGUI).

14 Type **EcDpPrStartQaMonitorGUI *mode ApplicationID &*** then press **Return/Enter** to launch the **QA Monitor** GUI.
The **QA Monitor** GUI (Figure 15-2) is displayed.

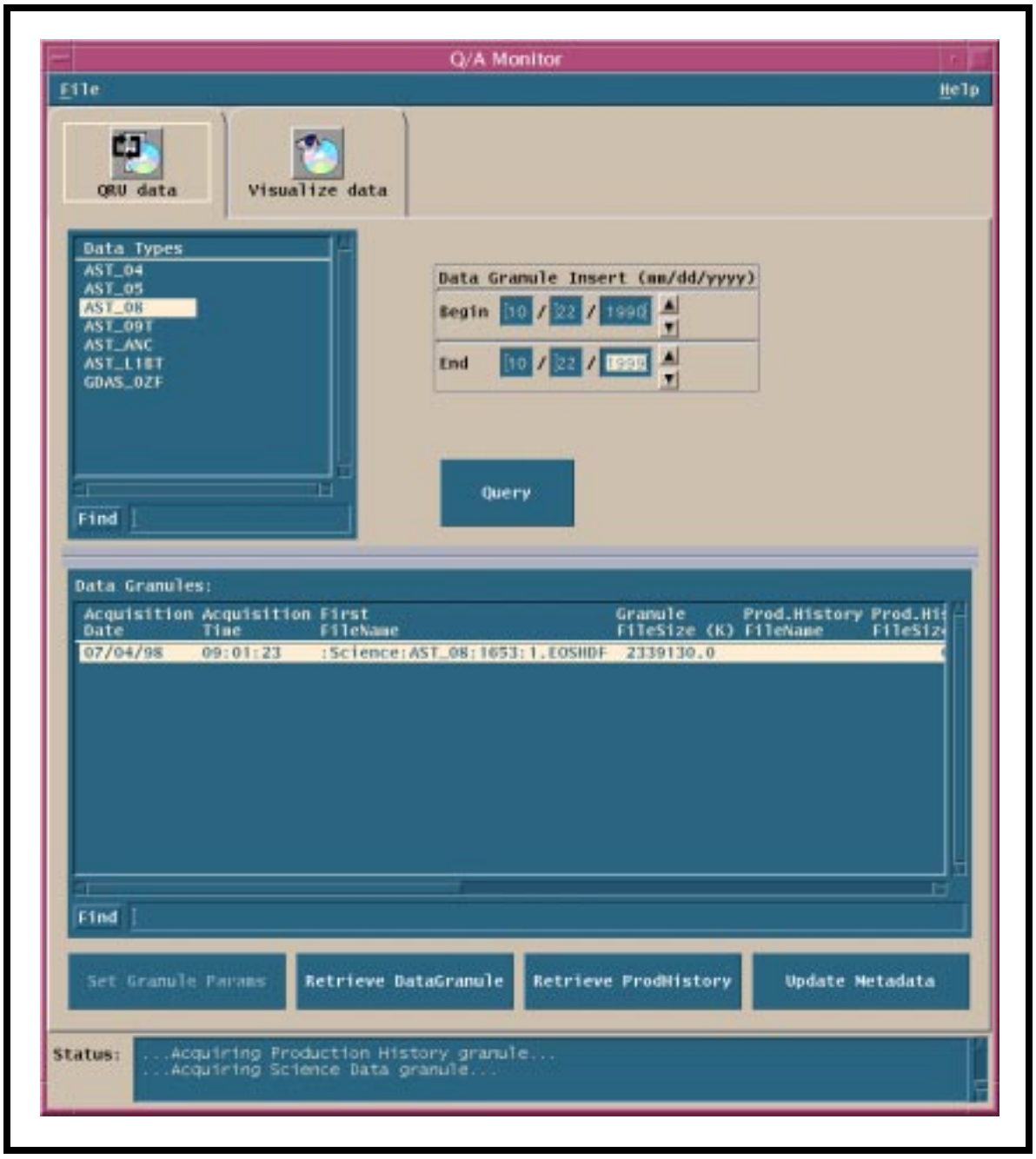

Figure 15-2. QA Monitor GUI - QRU Data Tab

15.1.1 DAAC Product QA with the QA Monitor

The Product QA process begins with the QA Monitor Application. The DAAC operations personnel will Query, Retrieve, and Update (QRU) the selected product. The operator will then

retrieve those specific products and perform a visual check of those products using the Visualize Data option of QA Monitor (Figure 15-3).

Retrieval and Viewing of Data Granules

1 Start up the QA Monitor

2 Select QRU Data tab.

3 From list of data types, select the ESDT or compose a query in query window and click on the **Query** button.

4 Select a data granule by filename from the list and click on the **Retrieve Data Granule** button.
Quit the QA Monitor GUI.

5 To visualize the data, select the data granule as described above and click on the **Visualize Data** button.
Displays EOS View GUI (Figure 15-3)

6 Open a HDF product file from which metadata is to be viewed, select the **File→ Open** button from the main menu bar.
A **File Selection Dialog** window will open and the user should be able to select the appropriate directory and file to open.
Once the desired product file has been opened, the specific types of HDF objects in the file will be listed in the **Contents** window.

7 From the **Contents** window double-click on a particular HDF Object (Vgroup, SDS, etc.).
The structure of the HDF object will appear in a dialog window with buttons on the bottom portion of the window to view the data of the object itself.

8 Display the science data values of this particular HDF object by selecting the **Table** button to display the table data of the object.

9 View the attribute values of this particular HDF object by selecting the **File-Attribute** button.
Metadata is referred to as **attribute data.**
Any metadata associated with the object will be displayed in another text window.

10 Quit when done by typing Q then press the **Enter** key.

Visualize Data

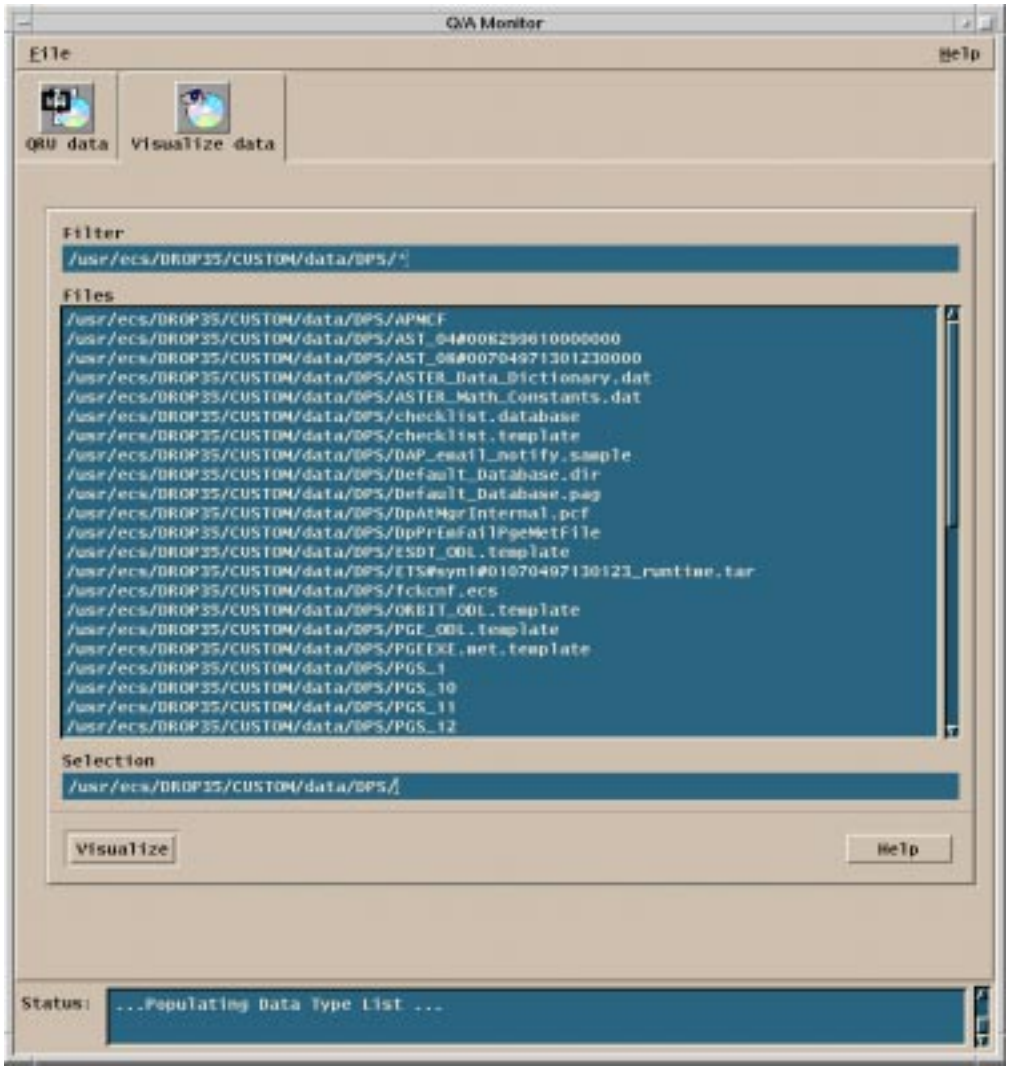

Figure 15-3. QA Monitor - Visualize Data

15.1.2 Updating QA Metadata

After the viewing, the operator will update the Operational QA flag for that specific product. The operator also updates the Science QA flags in response to an email request from SCF personnel, who have the responsibility for performing QA of their own products.

This procedure for updating QA metadata starts with the assumption that all applicable servers are currently running and the **QA Monitor** GUI **QRU data** tab (Figure 15-4) is being displayed.

Table 15.1-4 summarizes the QA metadata attributes and their descriptions.

Table 15.1-4. QA Metadata Attributes

Field Name	Data Type	Description
OperationalQualityFlag ScienceQualityFlag	character	DAAC and SCF quality status setting of a data granule parameter, selected by the user. The valid values are: - passed - failed - being investigated - not investigated - inferred passed - inferred failed
OperationalQualityFlagExplanation ScienceQualityFlagExplanation	character	Text describing quality status (less than 255 characters), input by user.
AutomaticQualityFlag	character	DAAC and SCF quality status setting of a data granule parameter, set during data processing.
AutomaticQualityFlagExplanation	character	Text describing quality status of a data qranule parameter - set during data processing.

Updating Quality Assurance (QA) Metadata using the QA Monitor

1 In the **Data Types** field, click on the data type to be checked.
It may be necessary to scroll through the **Data Types** list.
The selected data type is highlighted.
Only one data type can be selected at a time.
Alternatively, the **Find** field and button can be used for specifying a data type.

— The **Find** field is case-sensitive.

2 Click in the appropriate **Data Granule Insert** window field(s) and either type or use the up/down arrow buttons to enter the **Begin** date and **End** date in ***MM/DD/YYYY*** format.
In the **Data Granule Insert** window it is necessary to specify the range of dates (between the **Begin** date and the **End** date) to formulate a query for searching for the desired granule(s) to be checked. Time is based upon day of insert into the data server. If no dates are entered, an error message is displayed. The up and down arrows next to the duration fields may be used for modifying entries in each field.
The **Tab** key may be used to move from field to field.

3 Click on the **Query** button.
Granules within the specified date range appear in the **Data Granules** field.

4 In the **Data Granules** field, click on the granule for which metadata is to be updated.
It may be necessary to scroll through the list of granules.
The selected granule is highlighted.
Alternatively, the **Find** field and button may be used for specifying a data granule.

— The **Find** field is case-sensitive.

5 Click on the **Update Metadata** button.
The Update Metadata window is displayed.
The Update Metadata window displays one line for each parameter for the selected granule.

6 Click on a parameter in the Update Metadata window.
The selected parameter is highlighted.
The Edit Parameter dialog box is displayed.

7 Click and hold on the **Operator Quality Flag** option button, move the mouse cursor to the desired selection (highlighting it), then release the mouse button.
The selected metadata flag is displayed on the **Operator Quality Flag** option button.
The following options are available:

- **passed**
- **failed**
- **being investigated**
- **not investigated**
- **inferred passed**
- **inferred failed**

8 Type an explanation for changing the QA flag value in the **Explanation** field.

9 Click and hold on the **SCF Quality Flag** option button, move the mouse cursor to the desired selection (highlighting it), then release the mouse button.
The selected metadata flag is displayed on the **SCF Quality Flag** option button.
The same options are available as those on the **Operator Quality Flag** option button.

10 Type an explanation of the QA flag selection in the **Explanation** field.

11 Click on the **OK** button to accept the QA flag settings.
The Edit Parameter dialog box is dismissed.

12 To verify that the QA flag settings have actually been applied to the granule, first repeat Steps 1 through 5 to retrieve the same granule.
The **Granule Parameters** window (Figure 15-4) is displayed.
The QA flag values and explanations entered using the Edit Parameter dialog box are displayed.

13 Repeat steps as necessary to review additional granules.

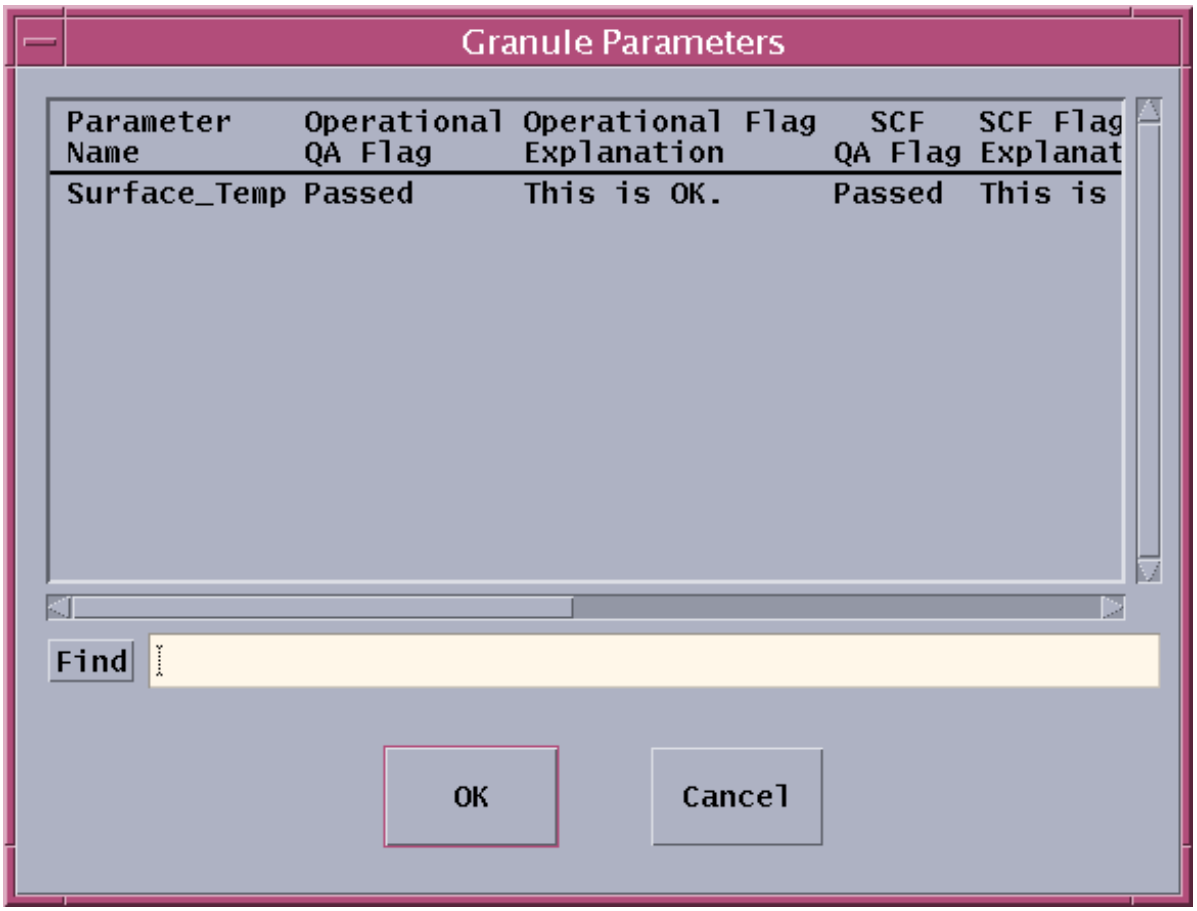

Figure 15-4. QA Monitor Granule Parameters Window

15.1.3 Production History

The Production History (PH) is created during PGE execution within the PDPS and then Inserted into the Data Server upon PGE completion. Included in the PH are the PGE log files. Accessing a Production History associated with a particular PGE run requires the DPR ID of the PGE run.

The Production History is retrieved using the QA Monitor GUI, using the following procedure.

Follow the procedures above for selecting a data granule.

- Select the **Retrieve ProdHistory** button and view the contents of the **Production History Log**.

15.2 QA Metadata Update Tool

The purpose of the QA Metadata Update Tool (QAMUT) is to enable Science Computing Facility (SCF) and Distributed Active Archive Center (DAAC) QA experts to modify values of their respective quality flags (i.e., ScienceQualityFlag and OperationalQualityFlag) on core

metadata, provided via a client, for multiple granules at a time in a batch mode (vs. one granule at a time using the QA Monitor).

The QA MUT and Logical ID/UR Report Generator Tool are being implemented as a DAAC Unique Extension to ECS per Engineering Support Directive #70. This work is not yet completed, but the reference version is available at the DAACs as explained below.

The QAMUT tool itself consists of two major components: The SCF component (for updating ScienceQualityFlags) and the DAAC component (for updating Operational Quality Flags). The procedures for using QAMUT are fully documented in the following text.

INSTRUCTIONS and GUIDE ON USING THE ECS DATABASE QA UPDATER 1.3

(by Richard H. Buss Jr. on May 1999 at GSFC. Revised: October 1999)
(GDAAC specific instructions are marked with "at GSFC" or "At GSFC")

TABLE OF CONTENTS

1. PURPOSE of the DATABASE QA UPDATER TOOL

The Database QA Updater is designed to update the values of the Quality Assurance (QA) Flags in the ECS Inventory metadata within the Science Data Server. The QA Updater tool updates either Science or Operational metadata QA Flags, but not Automatic metadata QA Flags. The QA Updater sets QA values for a granule containing one or more Measured Parameters that have been examined by science or DAAC facility staff for Quality Assurance and that need to have their QA Flags values and associated QA Explanations updated.

During one run, the tool can update multiple granules at a time by assigning them distinct QA values for each Measured Parameter. This is the main purpose and strength of the tool: updating batches of various granules at a time. The granules can contain different Measured Parameters, and the QA Update Tool will still update each of the QA Flags in each Measured Parameter for all the granules listed in the input. Similar updates for the same granule but with different Measured Parameters should be grouped contiguously on separate lines in the Request so that all the updates for a granule can be done at the same time.

A uniformly formatted input request is expected to have arrived at the DAAC by email, containing an email header with the requester's return address, and containing a list of the granules to be updated, along with the new QA Flag values for the specified Measured Parameters. This list of Updates can then be passed to the Updater tool through UNIX standard input (file or terminal input). A useful but configurable Cshell script is also provided to automatically setup the UNIX environment and to run the Updater automatically to process multiple requests that reside in a single directory.

2. SCOPE

The program reads a list of granule URs (Universal References; i.e. database Identifiers), and their Measured Parameters, along with the new QA Flag values and associated QA Explanations. The QA tool records the date of the Update at the end of the Explanations and then updates the QA values in the Inventory metadata database (not Archive Metadata), for each granule Measured Parameter. The program accomplishes this by overwriting previous QA Flag values and previous QA Explanations with the new QA Flags and new QA Explanations for each specified Measured Parameter. Thus, the QA Updater writes new QA metadata values but does NOT append QA values to existing metadata. Instead, blank inputs for QA Flag and Explanation values will cause the tool to set metadata QA values to "NULL" in the database, effectively erasing previous QA metadata.

Also, if the input specifies "ALL" for the Measured Parameter of a single granule, the tool will update all the Measured Parameter QA Flags in that granule so that all the flags will have the same value and all the QA explanations will have another uniform value. IMPORTANT: The tool can not update granules that have no Measured Parameters. Also, if a Measured Parameter is incorrectly specified for one granule, the Updater tool will perform updates to all other granules in the request, but it will not update any of the QA Flags in any Measured Parameters for that erroneously specified granule.

After performing the updates, the Database QA Updater will write to disk a file logging 1) the initial request, 2) the old QA values, and 3) the new updated QA values. This QA Update log file is intended to be saved as an record of the updates. In addition, the program will write runtime diagnostic messages that can be viewed on the terminal or redirected to a runtime log file (UNIX standard out). Finally, the program writes a temporary file specifying the status of the requests, notifies the requester by emailing this file back to the remote return address, and then the program deletes the notice from the local disk. This completes the QA Update process. The program must be invoked again in order for another request to be processed. However, if multiple requests reside in a single directory, the companion Cshell script, QAupdate.csh, will process all the requests together by invoking the QA Updater Tool multiple times.

3. TO INSTALL the TOOL

The core of the tool is a UNIX C++ binary executable that must run with its ascii configuration file. For
interactivity, there is an optional Graphical User Interface program in TCL language that can invoke the C++ executable. These components of the Database QA Updater are distributed as a package tar file and a README file.

a) Files∞
First, if needed, unpack the tar file on the ECS Science Data Server Machine (SDSRV) in your local QA
management directory with UNIX command:

tar -xvf packet_QA_UPDATER.tar .
*chmod u+x ClMuQaMetadataUpdate**
more ClMuQaMetadataUpdate.CFG

The account containing this directory must be able to receive email or files that contain the QA Update Requests. Your directory should now have the following files:

Filename	Purpose
README_QA_Updater	The master usage guide to the QA Updater
ClMuQaMetadataUpdate.CFG	The text configuration file for use in ECS
ClMuQaMetadataUpdate	The binary executable that updates the QA
ClMuQaMetadataUpdate.tcl	A manual Graphical User Interface to the Updater
ClMuQaMetadataUpdate.C	The partial source C++ code of the main program
QAupdate.csh	Cshell script to setup and run Updater on multiple requests
Input_example_QA_Updater	A sample input file that the Updater can process
Log_example_QA_Updater	A sample output log file that the Updater writes
Runtime_example_QA_Updater	A sample runtime diagnostic to standard out
Output_example_QA_Updater	A sample email notice sent back to the requester

The approximate sizes in kbytes of the Update Tool files are:

README_QA_Updater	28
ClMuQaMetadataUpdate.CFG	0.8
ClMuQaMetadataUpdate	3679
ClMuQaMetadataUpdate.tcl	10.6
ClMuQaMetadataUpdate.C	27
QAupdate.csh	1.1
Input_example_QA_Updater	1.6
Log_example_QA_Updater	9
Runtime_example_QA_Updater	0.6
Output_example_QA_Updater	1.7

It is recommended that all the ClMuQaMetadataUpdate* files, including examples be kept together in one directory at all times. It is also recommended that the real input requests be placed in a separate directory and that the output log files be placed in yet other separate directories. This structure should help you manage separately the inputs, the tool, and the outputs. Directories at GSFC:
\bin
\requests
\logs
\archiveQAupdates

b) Platform
The Database QA Updater C++ source code was compiled with CC compiler 4.0 at Raytheon Landover, MD in the ECS build environment containing links to other source code libraries such as Roguewave. It therefore can not be compiled at your site. The version and other information is imbedded in the binary file. This info can be view by the unix command:

what ClMuQaMetadataUpdate | more

The QA Updater Version 1.3 was compiled on a UNIX machine:

SunOS starfire 5.5.1 Generic_103640-08 sun4m sparc SUNW,SPARCstation-20.

The source code of the main program is given to help you understand the actions of the program. The program was successfully repeatedly tested on a

SunOS g0acs03 5.5.1 Generic_103640-20 sun4u sparc SUNW,Ultra-Enterprise.

at GSFC using input emails sent by UNIX mail, Netscape Messenger, Eudora, and Microsoft Outlook, all saved to disk with Netscape 4.0.7 Communicator.
****The average execution time is about 2 CPU seconds for each update.****

4. CONFIGURATION FILE

At the moment, the Database QA Updater is NOT formally integrated into ECS, so many lines in the configuration file are not yet used by ECS nor the tool.

There are, however, three required lines and one recommended one (Site) for you to specify correctly. At GSFC some entries from the ClMuQaMetadataUpdate.CFG file are:

```
EcDsScienceDataServerG1          = G1
Site                                  = GSFC
Name                                 = ClMuQaMetadataUpdate
STATUSFILEPATH                  = /home/quality/archiveQAupdates/
```

These are the names of the DCE SDSRV Database Cell, the DAAC, the QA Updater program, and the path to where the QA Update log files will be saved (write permission required). In addition, the line

```
DataFileName          = /usr/ecs/TS2/CUSTOM/security/EcSeRandomDataFile
```

can apparently be used in the future for security. The major and minor versions in the config file should be kept current so that they reflect the version of the binary tool itself: e.g. 1.3 .

The directory of the executable QA update tool is the default directory that the tool uses in case the configuration file can not be found by the tool. However, a configuration file can be placed in and referenced to an arbitrary location (neither links nor cross-mounting recommended). At GSFC, the /tools/gdaac/$MODE/bin/DSS and /tools/gdaac/$MODE/cfg/DSS contain the QA Updater Tool and configuration file, respectively.

5. TO RUN the TOOL

SHORT METHOD:
1) edit the QAupdate.csh script to refer to your local directories
2) place QA request files with valid URs into the requests directory
3) setenv MODE <TS1/TS2/OPS> (one of the three! Default: OPS)
4) execute QAupdate.csh on the UNIX command line within the Cshell

The ESDTs for the granules must already have been loaded by system administration into the Science Data Server and the Science Data Server must already be running. If the tool becomes formally part of ECS, an initial dce_login will probably be required, too, before the tool can run. The QA Update Tool is known to interact with the ECS system by:
A) updating the SDSRV database metadata,
B) querying ESDTs installed in SDSRV, and
C) fulfilling subscription events to metadata updates.
Since the QA updates are restricted to the SDSRV, the tool is simple to run and will work in parallel with other ECS activities such as INGEST.

All DAACs can run the Updater on their local file structure. (However, at GSFC, the DAAC Database QA Updater can be run on g0acs03 from the quality account (/home/quality/bin) with the Tool and configuration file in /tools/gdaac/$MODE/bin/DSS and /tools/gdaac/$MODE/cfg/DSS .)

LONG METHOD:
---First, set the MODE to one of the following three:
setenv MODE <TS1/TS2/OPS>

---Second, source the setup file
source /usr/ecs/$MODE/CUSTOM/utilities/EcCoEnvCsh

IMPORTANT!!!!!!!!If this file is missing, manually convert the EcCoEnvKsh file in that directory to EcCoEnvCsh and then source the converted file.

This step creates links to needed libraries through the LD_LIBRARY_PATH path. Some DSS and CSS subsystem libraries are probably both used by the tool. (In an emergency at GSFC, source EcCoEnvCshcopy in /home/quality/bin.)

---Third, invoke the Database QA Updater. The tool can be run two ways: by command or by GUI window.
1) From the command line, an example execution (at GSFC, tool is in /tools/gdaac/<MODE>/bin/DSS): ClMuQaMetadataUpdate ConfigFile ClMuQaMetadataUpdate.CFG ecs_mode $MODE < QAupdate_emailrequest.txt

where the format is:
executable ConfigFile (Configuration filename) ecs_mode (MODE) < (input filename) > (output runtime log filename)

and where the "> output runtime log filename" is optional to save STDOUT. Note that absolute and relative path names can be used for the filenames on the command line. Note also that the optional runtime log file is distinctly different from the mandatory QA update log filename specified in the configuration file; in the first case, program runtime information is saved or displayed; in the second case, the QA metadata values and Update Request are saved.

2) An operator can also invoke and use a graphical tool ClMuQaMetadataUpdate.tcl to perform QA updates with GUI buttons.
Example:
ClMuQaMetadataUpdate.tcl ClMuQaMetadataUpdate.CFG $MODE &

Limited online help for running the Updater is available in the GUI. STDOUT also is written in the GUI window but can only be saved manually to a file.

6. INPUT FILE FORMAT (Specifications)

Though the wrapper script QAupdate.csh can process multiple request files in one directory, the Updater tool itself processes one QA update request file at a time. The input file is constructed by the requester who has examined the granules for quality and has decided to update the QA Flags in the granules. The requester should have previously obtained a granule identifier (UR, Universal Reference) from ECS, for example, by the subscription notices that are emailed to the requester when data is transferred by ECS to the requester. (Alternately, to facilitate identification of data granules at GSFC, the UR is published together with the local granule ID in a lookup file. Note that the UR is NOT contained inside the granule itself nor in its metadata!) In any case, the Update Request must specify the granule by UR for the Updater tool to work.

The input QA Update request file must have four major items:

email header
begin line
request line(s)
end line

Sample Input File (Example):

Date: Wed, 25 Nov 1998 12:55:42 -0500
From: rbuss@gsfcsrvr4.gsfcmo.ecs.nasa.gov
Reply-To: Richard.Buss@gsfc.nasa.gov
Organization: EOS-DAAC GSFC
X-Mailer: Mozilla 4.07 [en] (X11; I; SunOS 5.5.1 sun4m)
MIME-Version: 1.0
To: quality@g0ins01u.ecs.nasa.gov
Subject: QA Update Request for ECS. This is a test of the DAAC Database QA Update Tool.

begin QAMetadataUpdate Science
UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:19:SC:AST_09T.001:1988 GuruTej Passed test

UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:18:SC:AST_05.001:1990 GuruTej Passed test
end QAMetadataUpdate

This concludes the Example input file. Details on each item follow:

1) The email header MUST contain at least three lines in any order:
From:
Subject:
To:

The email header safely can contain many other fields such as Reply-To: or Received: or Date:, but it must contain at minimum, these three fields.

A) The Database QA Updater will email the return notice back to the sender at the Reply-To: address, if it exists before the end of the header, or alternatively, back to the From: address.

B) The Subject line is a valuable means of identifying the QA update Request, both for the requester and for the DAAC. At GSFC the subject line should begin with
Subject: QA Update Request for ECS.

and then followed on the same line by any other information that the requester wants to use to identify the request. The QA Updater will echo back this subject line in the return email notice sent to the requester. This will help identify which Update request was actually completed. The use of single quotes (') in the subject line is, however, strongly discouraged to avoid confusing email programs.

C) The To: address that receives the requests at GSFC is quality@g0ins01u.ecs.nasa.gov .

2) a begin line with either Science or Operational specified for the kind of update:
begin QAMetadataUpdate Science
or
begin QAMetadataUpdate Operational

The request must be for a single type of update; it can not contain both kinds of update at once.

3) one or more update entries, one entry to each line and four fields in each entry, each separated by tabs, not blanks nor commas, nor double quotes. The fields on one line are
A) Database ID name/number (UR) B) Parameter name C) QA Flag Value D) QA Flag Explanation.
FORMAT:
NAME:NUMBER<tab>PARAMATER<tab>QAFLAG<tab>Explanation
Example:
UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:18:SC:AST_05.001:1990
GuruTej Not Investigated An Explanation

The Updater has been successfully tested on a request with 63 update UR lines.

4) an end line with a return (new line) as the last character:
end QAMetadataUpdate

*Here are some **IMPORTANT** details to consider when the requester creates an input file:*

--When a cut and paste is used to send in a request, it often loses the crucial tab and line-wrap formattings. It is recommended that an editor be used to compose the Request, rather than pasting into an email messaging utility. The edited file can then be imported or read into the emailer rather than a dangerous cut and paste.

---The six valid QA flag values for both Science and Operational in the 1999 ECS Data Model are:
i) Not Investigated ii) Being Investigated iii) Passed iv) Failed v) Inferred Passed vi) Inferred Failed. No double
quotes (") can be with the Flag or the request will fail.

---The Explanation field is free form, can be blank, and can contain special characters EXCEPT double quotes (") or tabs. The length of the Explanation can be up to 235 characters. An update date is always appended to the Explanation (for a maximum length of 255 characters) when the database is updated.

---The same granule can be updated twice within the same request, but the entry lines for that granule must
immediately follow each other (be contiguous). Otherwise, the program will crash (bus error) when the granule is accessed for the second update attempt! This applies particularly to granules with several Measured Parameters.

---For a given update type (either Science or Operational), the requester can specify ALL as the parameter name in order for all Measured Parameters in the granule to be set with the same value and with the same explanations (but values and explanations separate from each other.)

7. OUTPUTS

Each time it runs, the QA Updater program creates several outputs:
1) New QA Metadata in the Science Data Server Database Table (DsMdMeasuredParamater).
2) An Update log file, named by current date and time, containing the
  A) address of the requester, the request subject, and a copy of the request
  B) QA Metadata values before and after the update
  C) runtime status messages and success/fail summary
3) A return status notice that is emailed back to the requester. It contains:
  A) a short summary, if the QA Updater was successful
  B) a long description with error/warnings for most failures.
4) An output stream to Standard Out (STDOUT), listing the progress of the program.

(The QAupdate.csh wrapper script sends a few minor diagnostic messages to STDOUT (the screen).

8. CAPABILITIES OF the DATABASE QA UPDATER

The tool:
1) Updates Operational and Science QA metadata separately.
2) Reads a single request file containing an email header and one or more granule ID updates.
3) Successfully batch updates all specified QA metadata in those granules.
4) Updates together both the QA Flag and the QA Explanation.
5) Completely updates the QA metadata including those with null (blank) values.
6) Writes the date of the Update at the end of the QA Explanation.
7) Successfully processes one update request after another.
8) Works if multiple simultaneous QA update processes are invoked.

9) Accepts full path and current directory file names (known to work on 143 characters).
10) Updates other granules if Measured Parameter is incorrectly specified for some granules.
11) Successfully processes requests with blank lines or one line containing only tabs.
12) If requested, updates other Measured Parameters if one is incorrect for the same granule.
13) Will update the QA flag in all Measured Parameters in the granule if ALL option is specified.
14) Processes QA explanations with special characters such as @ and ' and one extra <tab>.
15) Sends return email to most REPLY-TO: addresses instead of FROM: address.
16) Writes various success, warning and error status messages to Standard Out (STDOUT).
17) Writes similar status messages to the Update log file and to the return notice.
18) Deletes the return notice from disk after it has been emailed back to the requester.
19) Can be used with a wrapper script QAupdate.csh to process multiple request files.

9. LIMITATIONS OF the DATABASE QA UPDATER

1) Issues warnings in the log file ("is invalid with respect to rule") that invalid metadata was requested, but still allows the values to be written into the database!
2) Allows further QA updates of the same granule in the same message only if the update request lines for that granule are contiguous (sequential) in the request file. Otherwise, if the same granule is requested to be updated again in the same request file, and the update lines are separated in the file, then the program crashes!
3) Does not process QA Explanations nor Flags containing double quotes "
(error written in log: "Could not update metadata in catalog")
4) For safety reasons to the Database QA Metadata, requests that appear to be corrupted and do not follow the format, are not processed, other than to send error messages to the log file and back to the requester. NONE of the granules are updated for these format/content errors, which include:
   A) extra entries (total >4) on any line ("Too many fields on line" message is written to the log)
   B) a non-existent database ID name/number (UR) on any line, including a blank or misspelled UR
   C) missing begin line
   D) missing or incomplete email header
5) The program exits abruptly, without warning messages, if the specified configuration file is misspelled or nonexistent.

10. FUTURE

A more complex Cshell wrapper script is under development at GSFC DAAC. The script sorts the URs to prevent crashing caused by repeating URs, provides a configurable security limit of, say, 10,000 allowed updates per message, and checks metadata valids before the Updater is run. The wrapper script also tests for the presence of the configuration file and will exit with warning messages if the file is not found. It also will manage the requests by filing them into failed, passed, and partially successful categories, so that appropriate actions can be taken by the DAAC and the requester.

11. ACKNOWLEDGEMENTS

The author appreciates the advice, consultation, and assistance of Robert Kummerer, A. K. Sharma, Sushma Singhal, and Mark Fuerst during the development and testing of this software. I

am grateful for the expertise of Robert, the collaboration of A. K., the design and conceptualization of Sushma, and the support of Mark. Thanks to all the programmers who contributed to the tool and to NASA GSFC EOSDIS with Robert Lutz for making Quality Assurance possible in the ECS metadata.

If you notice any inconsistencies or errors in this guide to the Database QA Updater, you can email them to Richard.Buss@gsfc.nasa.gov or the GSFC DAAC Code 902.2 Greenbelt MD 20771 for corrections.

12. APPENDIX: ERROR AND WARNING MESSAGES

To access the status messages written by the program, read the Standard Out stream. This stream contains some status messages, as well as contains the name of the associated Updater log file (named with a date/time stamp). Print the log file to examine more status messages. The log file also contains the address of the requester and the subject of the request, which identify which log file corresponds to each processed request.

1) Operating system statuses

---Warning: Could not open message catalog "oodce.cat"

You can safely ignore this ECS warning message produced by the dce utility.

---ld.so.1: ClMuQaMetadataUpdate: fatal: libDsClSh.so: can't open file: errno=2
Killed

This is a failure to link with the required libraries.
Make sure you are on the SDSRV machine, at GSFC g0acs03.
Try sourcing the EcCoEnvCsh setup file and re-running the Updater.

---ld.so.1: ClMuQaMetadataUpdate: fatal: relocation error: symbol not found:
__0oPGlParameterBasectPCc: referenced in ClMuQaMetadataUpdate
Killed

This is a failure to link with the required libraries probably
because of a mismatch between the tool version and the ECS drop version.
A new drop version of ECS installed at your site might have caused this error.
The executable needs to be recompiled at Landover/Raytheon under a newly
created clearcase view for that drop version.

---Permission denied

Try adding execute (chmod +x) to the file permissions for ClMuQaMetadataUpdate.

2) Standard Out messages

---SUCCESSFULLY updated all granules.
Errcount = 0

---ERRORS occurred during processing the request. QA Update program is quitting.
NO GRANULES WERE UPDATED FOR THIS REQUEST.

(Probably one or more UR granule IDs in the request are not in the database.)

---WARNING: Invalid metadata was entered in the Inventory.

(The requester misspelled the value of QA Flags. Process a revised request
containing valid QAFlag values; otherwise, bad metadata will remain in the database!!)

---ERROR in updating QA: Email return address is missing from request.
NO QA updates were performed for this request.

(The email request was saved only partially to disk before the program ran.)

---ERROR: Update Request had invalid metadata values such as quotes.

(No double quotes are allowed in the QAFlag and QA Explanations in the request.)

---WARNING: A measured parameter did not initially have any QA Flag.

(Best to check that the database contains updated QA values. If, not, try
processing the request again; it might work the second time.)

3) Updater program status messages:

==========
---Error creating DsClESDTReference for granule
Failure to create ESDT Reference, QA Update program is quitting.
NO GRANULES WERE UPDATED FOR THIS REQUEST.

The database ID (UR) is missing or wrong for AT LEAST one entry in the input request.
This happens when:
a) an ESDT specified in a request UR does not match the actual ESDT UR (database ID) for a particular granule. (E.g. the requested UR contains MOD01 when the actual granule is MOD07_L2 type.)
b) a UR field is missing from any one line in the input request.
c) one UR Database ID in the request is nonexistent in the SDSRV database. (E.g. 44790 when there are only 30000 granules in the database.)
d) there are only blank lines between the begin and end lines in the request,

In these cases, NO updates on ANY granules have been done; a revised request for updating the same granules, but all with proper URs, will have to be submitted by the requester.

==========
---*****MeasuredParameter <param> not found in granule metadata

Errors during update process. Updates not done for this granule UR.
An ERROR occurred during the update process for <granule>
**** This QA Update was not done for granule: <granule>
Other granules will be updated.

A line in the input request file has a wrong or missing parameter name. The program does not make any of the QA
update for this granule. The program makes all other updates for the other granules, though. The program automatically by email notifies the requester of this error.

==========
---UnnamedPL[(MeasuredParameter not found.) ErrorMsg(ESDT Execution failed)]
**** This QA Update was not done for granule: <granule>
Other granules will be updated.

The granule does not even have a Measured Parameter or QA Flag to be updated!
This granule should be deleted from the Update request because the metadata do not permit updating or even setting the QA Flags.

==========
---ESDTResults[(Could not update metadata in catalog) ErrorMsg(ESDT Execution failed)]
ESDTStatus(0)
CmdSuccess(1)]]

The request line has double quotes in the explanation.

==========
---Can't get status file path from config file.

Make sure that the directory path to the logfile, as specified in the configuration file, is spelled correctly and has write permission.

==========
---Unable to find a begin line or the email header is incomplete.
Input file does not have proper format: Subject, From, To, begin.
NO QA updates were performed for this request.

Check the email header to make sure all three required lines are present.
Check that the begin line has Science or Operational after the begin word.
The program should be run again on the corrected request.

==========

--Too many fields on line
End of input requests
Errors occurred during processing, quitting.

There are extra fields on one input line (5 or more filled out fields separated by tabs).
The program does NOT process any of QA updates on ANY of the granules.
Notify requester that the request did not have the proper format.

==========
---WARNING: end QAMetadataUpdate line not found.
Processing will continue, but lines may be missing.
End of input requests

Check that the input request file was not split or the latter part deleted, because the input file does not have an end line (or is missing a <return> after the end line). If the request was erroneous, notify the requester.

The program finishes and does all the updates listed in the processed request, but some of the updates in original request email might not have been processed because the request was inadvertently altered.

==========
---ESDTResults[WARNING(Attribute
(INVENTORYMETADATA:MeasuredParameter:MeasuredParameterContainer:QAFlags:Scien
ceQualityFlag=Fail)

is invalid with respect to rule [Match(Passed,Failed,Being Investigated,Not Investigated,Inferred
Passed,Inferred Failed)])
UR(UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:19:SC:AST_09T.00
1:1076)]
ESDTStatus(1) CmdSuccess(1)]]

The request updates QA flags with invalid new metadata values. The program processes all entries, finishes, but writes this warning.
In the log, both ESDTStatus(1) CmdSuccess(1) mean success; either ESDTStatus(0) or CmdSuccess(0) for a given request line means that the particular granule QA update failed.

==========
---ESDTResults[WARNING(Missing parameter
INVENTORYMETADATA:MeasuredParameter:MeasuredParameterContainer:QAFlags:Autom
aticQualityFlagExplanation
in group
INVENTORYMETADATA:MeasuredParameter:MeasuredParameterContainer:QAFlags.)

The metadata of this granule were initially incorrect before the QA Updater was even run.
The MCF file or the ESDT in the Inventory SDSRV database is not properly written to receive QA metadata updates. Before the QA update request can be processed, the MCF file or ESDT for the data product will have to be modified so that the PGE can write at least one QA metadata Flag (e.g. AutomaticQualityFlag) in the metadata.

==========
---ERROR on begin QAMetadataUpdate line: expecting \"Science\" or \"Operational\""

The type of request is misspelled or missing on the begin line of the request.
---

13. APPENDIX: BUILD LINE

```
      CC -I. -I./sun5.5 -I/tools/oodce/usr/include/oodce
-I/tools/oodce/usr/include/ -I/usr/include/dce -I/usr/include
-I/ecs/formal/CLS/include -I/ecs/formal/COMMON/include
-I/ecs/formal/COMMON/include/sun5.5      -I/ecs/formal/CSS/include
-I/ecs/formal/CSS/include/sun5.5  -I/ecs/formal/MSS/COTS/peer/src/include
-I/ecs/formal/MSS/include -I/ecs/formal/MSS/include/sun5.5
-I/ecs/formal/IOS/include -I/ecs/formal/IOS/include/sun5.5
-I/ecs/formal/DSS/include -I/ecs/formal/DSS/include/sun5.5  -g
-DSUNOS55X  -ptrsun5.5/Templates/ClMuQaMetadataUpdate.o           -mt
-I/ecs/formal/IOS/include/ -I/ecs/formal/IOS/include/sun5.5
-I/tools/rogue70_new -DRW_MULTI_THREAD -D_REENTRANT -DRWDEBUG=1
-I/tools/rogue70_new -DRW_MULTI_THREAD -D_REENTRANT -DRWDEBUG=1
-I/tools/sybOCv11.1.0//include  -Dunix -DNIDL_PROTOTYPES -D_REENTRANT
-D_CMA_PROTO_ -DSYSV -I/tools/oodce/usr/include/oodce -I/usr/include/dce
-I/tools/oodce/usr/include/oodce -I/usr/include/dce   -mt -c -o
sun5.5/ClMuQaMetadataUpdate.o ClMuQaMetadataUpdate.C
"/ecs/formal/CSS/include/EcPfClient.h", line 74: Warning:
EcPfClient::PfProcessEvent hides the virtual function
EcPfGenProcess::PfProcessEvent(EcAgEvent*, EcTAgLogType).
"ClMuQaMetadataUpdate.C", line 542: Warning (Anachronism): Undefined
character escape sequence "\-".
"ClMuQaMetadataUpdate.C", line 542: Note: Type "CC -migration" for more on
anachronisms.
"ClMuQaMetadataUpdate.C", line 542: Warning (Anachronism): Undefined
character escape sequence "\-".
3 Warning(s) detected.
      /ecs/formal/COMMON/CSCI_Util/src/Build//cmAccess
/ecs/formal/CLS/bin/sun5.5/ClMuQaMetadataUpdate
      /ecs/formal/COMMON/SysBuild/generate_EcsVersion_C
```

```
ClMuQaMetadataUpdate
        CC -DSUNOS55X -c
/ecs/formal/COMMON/SysBuild/Versions/ClMuQaMetadataUpdate_EcsVersion.C -o
/ecs/formal/COMMON/SysBuild/Versions/ClMuQaMetadataUpdate_EcsVersion.o
        CC -ptrsun5.5/Templates/Pr_ClMuQaMetadataUpdate
-ptrsun5.5/Templates/ClMuQaMetadataUpdate.o
-ptrsun5.5/Templates/ClMuQaMetadataUpdate.o     -R
../../lib/COM:../../lib/CLS:../../lib/DMS:../../lib/CSS:../../lib/DPS:../../lib/DSS:../../lib/INS:../../lib/I
OS:../../lib/MSS:../../lib/PLS:../../lib/V0_Client:../../../COTS/rogue/lib:../../../COTS/sybase/lib:../..
/../COTS/xaclient/lib:/ecs/formal/CLS/lib/sun5.5:/ecs/formal/COMMON/lib/sun5.5:/ecs/formal/
CSS/lib/sun5.5:/ecs/formal/MSS/lib/sun5.5:/ecs/formal/IOS/lib/sun5.5:/ecs/formal/DSS/lib/sun5.
5:/view/DROP5A_sun5.5/ecs/formal/COMMON/lib/sun5.5:/view/DROP5A_sun5.5/ecs/formal/
CSS/lib/sun5.5:/view/DROP5A_sun5.5/ecs/formal/MSS/lib/sun5.5:/view/DROP5A_sun5.5/ecs/f
ormal/IOS/lib/sun5.5:/view/DROP5A_sun5.5/ecs/formal/DSS/lib/sun5.5:/ecs/formal/IOS/lib/sun
5.5:/view/DROP5A_sun5.5/ecs/formal/IOS/lib/sun5.5:/tools/rogue70_jun99/lib/:/tools/rogue70_j
un99/lib:/tools/rogue70_jun99/lib:/tools/sybOCv11.1.0//lib:/tools/oodce_jun99/usr/lib
-o /ecs/formal/CLS/bin/sun5.5/ClMuQaMetadataUpdate
sun5.5/ClMuQaMetadataUpdate.o -L/ecs/formal/CLS/lib/sun5.5
/ecs/formal/COMMON/SysBuild/Versions/ClMuQaMetadataUpdate_EcsVersion.o
-L/ecs/formal/CLS/lib/sun5.5 -L/ecs/formal/COMMON/lib/sun5.5
-L/ecs/formal/CSS/lib/sun5.5 -L/ecs/formal/MSS/lib/sun5.5
-L/ecs/formal/IOS/lib/sun5.5 -L/ecs/formal/DSS/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/COMMON/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/CSS/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/MSS/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/IOS/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/DSS/lib/sun5.5      -lDsClSh -lDsCnSh
-lEcUtMiscSh -lDsShSh -lGlSh -L/ecs/formal/IOS/lib/sun5.5
-L/view/DROP5A_sun5.5/ecs/formal/IOS/lib/sun5.5 -lIoAdSearchSh -lIoAdSubsSh
-lIoAdCoreSh -lIoAdServerSh -lIoAdProxySh -lEcPoSh -lEcPoDbRWSh -lDsCnSh
-lEcUtMiscSh   -L/tools/rogue70_jun99/lib/ -lrwdbg_mt
-L/tools/rogue70_jun99/lib -lrwtoolg_mt -lGlSh -lEcUtMiscSh -lEcDcNotifySh
-lEcDcMsgPsng1 -lCsEmMailRelASh -lC -lEcNsServiceLocSh -lEcFhSh
-lEcNsServiceLocatorSh -lEcUrSh -lEcPfSh -lEcCfSh -lEcAgInstrm
-lagenteventSh -leventSh -lCsFtFTPSchedObjSh -lCsFtFTPRelASh -lexpect -ltcl
-lEcTiTimeSh -lEcSeSybSecurityStubbedSh -lEcSeUtilityCtStubbedSh
-lEcSeServerKeyMgmtSh -lEcSeCmiSh -lEcDcMsgPsng1Sh -lEcDnDirSh  -lEcUtSh
-lEcUtFactorySh -lEcPtSpecialLockSh -L/tools/rogue70_jun99/lib -lrwtoolg_mt
-Bstatic -L/tools/sybOCv11.1.0//lib /tools/sybOCv11.1.0//lib/libct_r.a
/tools/sybOCv11.1.0//lib/libcs_r.a /tools/sybOCv11.1.0//lib/libcomn_r.a
/tools/sybOCv11.1.0//lib/libintl_r.a /tools/sybOCv11.1.0//lib/libtcl_r.a
-Bdynamic /tools/sybOCv11.1.0//lib/libtli_r.so -L/tools/oodce_jun99/usr/lib
-loodce  -lnsl -Bdynamic  -ldce -lsocket -Bdynamic -lthread -lm -lresolv
/usr/lib/libintl.so -ldl -lm   -mt

clearmake: Warning: Configuration record will not include objects accessed using view
"DROP5A_sun5.5"
```

This page intentionally left blank.